REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-03-

0313

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE** FINAL	**3. DATES COVERED** *(From - To)* 01 Apr 01 - 31 Mar 03
4. TITLE AND SUBTITLE Modeling and Analysis of Information Attack in Computer Networks		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER F49620-01-1-0288
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) David L. Pepyne and Yu-Chi Ho		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Harvard University Cambridge, MA 02138		**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research AFOSR/NI 801 Randolph Street, Suite 732 Arlington, VA 22203-1977		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release, distribution unlimited.

20030818 001

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This is the final report of a three-year research project conducted under the AFOSR sponsored Critical Infrastructure Protection and Information Assurance Fellowship Program. The research focused on information networks and methods to understand their vulnerability to information-based attack (as opposed to physical and other forms of attack). Information based attacks are attacks that can be carried out from anywhere in the world, while sipping cappuccino at an Internet café or while enjoying the comfort of a living room armchair. Such attacks are particularly problematic because they take place in a "virtual cyber world" that lacks the social, economic, legal, and physical barriers and protections that control and limit crime in the material world. Research outcomes include basic theory, a modeling framework for Internet worms and email viruses, a sensor for user profiling, and a simple protocol for enhancing wireless security.

15. SUBJECT TERMS

Complex systems, network vulnerability modeling, malicious mobile code, intrusion detection, user profiling, wireless security

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** David Pepyne and Yu-Chi Ho
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED	UNCLASSIFIED	21	**19b. TELEPHONE NUMBER** *(include area code)*

MODELING AND ANALYSIS OF INFORMATION ATTACK IN COMPUTER NETWORKS

Sponsor: Department of the Air Force
Air Force Office of Scientific Research

Sponsor Award Number: F49620-01-1-0288

FINAL REPORT

01 April 2001 – 31 March 2003

Principal Investigator:

Yu-Chi Ho
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
PHONE: (617) 495-3992
FAX: (617) 496-6404
ho@hrl.harvard.edu

Fellow:

David L. Pepyne
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
PHONE: (617) 495-8911
FAX: (617) 495-9837
pepyne@hrl.harvard.edu

EXECUTIVE SUMMARY

This is the final report of a two-year (4/01/2001 – 3/31/2003) research project conducted under the AFOSR sponsored Critical Infrastructure Protection and Information Assurance Fellowship Program. The research focused on information networks and methods to understand their vulnerability to information-based attack (as opposed to physical and other forms of attack). Information based attacks are attacks that can be carried out from anywhere in the world, while sipping cappuccino at an Internet café or while enjoying the comfort of a living room armchair. Such attacks are particularly problematic because they take place in a "virtual cyber world" that lacks the social, economic, legal, and physical barriers and protections that control and limit crime in the material world. Research outcomes range from basic theory through to addressing vulnerabilities in specific networking technologies.

Basic Theory. Development of the F-matrix, a mathematical framework for studying the implications and impacts of complexity on engineering design and optimization.

Malicious Mobile Code. Development of a modeling and simulation framework for studying the cascading spread of malicious code such as the increasingly common Internet worms and email viruses.

User Profiling. Designed and evaluated a simple sensor for profiling and detecting abnormal and suspect user behavior—the first step to arresting cyber criminals.

Wireless Security Protocol. Development of a simple cryptographic protocol rooted in message authentication and demonstration of its application to securing inherently vulnerable wireless data links.

TABLE OF CONTENTS

1. INTRODUCTION

This is the final report of a two-year (4/01/2001 – 3/31/2003) research project conducted under the AFOSR sponsored Critical Infrastructure Protection and Information Assurance Fellowship Program. The broad goal of the project was to explore the development of a theory for understanding and assessing the vulnerability of information networks to 'information attack' so as to be able to choose strategies to prevent such attacks and to make our critical information networks more secure, predictable, and reliable. To this end, our research focused on the following topic areas:

- **Complexity**. By almost any measure of complexity (Lloyd, 2001), information networks are complex. Understanding the sources of complexity and the limitations that these sources impose on our ability to analyze, optimize, and secure an information network is a necessary first step towards discovering new ways to deal with complexity and its limitations.

- **Cascading Failure**: Internet worms and viruses are becoming an increasing cause for concern. While we have been fortunate in that the Internet worms and viruses have so far been largely benign, these malicious software programs have the potential to rapidly cascade through an information network like a disease through an animal population to cause widespread damage and disruption. Developing models to understand propagation mechanisms and how they are impacted by network topology, routing architecture, and response strategy is needed to design techniques to detect and defend against such cascading phenomena.

- **Information Classification**. A fundamental difficulty in large, distributed information networks is establishing the legitimacy of service requests. First, it is generally not possible to positively determine the identity of the user who submitted the request. The presentation of credentials such as username and password is no assurance of identity, since these can be easily guessed or stolen. Moreover, since information networks lack many of the social, economic, legal, and physical barriers and protections that control and limit crime in the material world, there is less to restrain people, even "trusted" insiders, from misusing an information network. Intrusion and misuse detection techniques able to classify user activities as normal or suspect are necessary to protect a system against illegitimate activities.

- **Technology Vulnerabilities**. There often seems to be an inverse relationship between the convenience of a technology and its security. Wireless technologies in particular are very convenient, but particularly vulnerable to attack. Specific protocols are needed to reduce this vulnerability. These protocols, however, must be carefully designed and implemented, as evidenced by the notorious weaknesses of the Wired Equivalent Privacy (WEP) security protocol used in the current and very widely deployed 802.11b, WiFi wireless networking devices. Many fixes have, of course, been proposed for WEP, but many of these are quite elaborate and will require significant changes to the current wireless infrastructure. The complexity of these proposed solutions, while they look good on paper, can work to the advantage of an attacker since their implementation is difficult and because their complete testing is generally not possible. We believe that good security does not necessarily require complex solutions, but should rather focus on simple security protocols.

We believe that significant progress was made on the above topics. The discoveries and understandings that arose from this project has lead to the identification of new and promising research directions that are being pursued by the PI, the Fellow (now a full-time Research Associate at Harvard University), and researchers at other universities that we collaborated with during the course of this project. Specific accomplishments include the following:

- The development of new theory for understanding fundamental limitations to optimization and fundamental relationships between complexity and system security (Ho and Pepyne, 2002; Ho, Zhao, and Pepyne, 2003).
- Initial development of a modeling framework for understanding worm and virus attacks (Pepyne, Gong, and Ho, 2001).
- Development and evaluation of new sensor technology for intrusion and misuse detection (Hu, Pepyne, and Gong, 2002).
- Invention of a new protocol for wireless data link layer security (Pepyne, Ho, and Zheng, 2003).

The remainder of this report is organized as follows. Section 2 provides additional details about the above accomplishments. In Section 3 lists the publications produced during the project. Section 4 gives a brief biographical sketch for the Fellow funded by the fellowship. Section 5 is a bibliography of the other references mentioned in this report.

2. BACKGROUND AND SUMMARY OF ACCOMPLISHMENTS

The following subsections provide a summary of accomplishments, including sufficient background to motivate and understand their significance.

2.1 Complexity

In (Ho and Pepyne, 2002; Ho, Zhao, and Pepyne, 2003) we developed a framework for understanding the basic limitations to our ability to optimize and secure complex systems such as today's information networks. This framework provides both descriptive understanding and suggests prescriptive design guidelines.

Starting from basic principles, the goal of much of engineering is optimization. We are generally seeking an engineering solution that is "best" in some sense, or at least provides an "improvement" over the current solution. The first step in setting up an optimization problem is defining the input space of solutions X and output space of performance values Y. Regarding these two spaces, we can make the following assumption:

- *Finite World Assumption.* The only general-purpose tool for studying complex systems such as large information networks is a simulation model run on a digital computer. Because the world of digital computers is a discrete and finite one, we assume that X and Y are discrete and finite spaces.

The *Finite World Assumption* leads immediately to a construct we call the "Fundamental matrix", or F-matrix for short. The rows of this matrix are the inputs from X (solutions, strategies, designs), the columns are problem instances (mappings from X to Y), and entry i,j (an element of Y) gives the performance of the row i solution on the column j problem instance. If there are $|X|$ total possible inputs and $|Y|$ total possible outputs, then there are $|F| = |Y|^{|X|}$ total possible unique problem instances in the set $F = \{f: X \to Y\}$. That is, an F-matrix has $|X|$ rows and $|F|=|Y|^{|X|}$ columns, and we immediately note that the number of columns is exponential in the number of rows. To fix ideas, an example F-matrix for $|X| = 3$ and $|Y| = \{0, 1\}$ is illustrated in Fig. 1. To simplify the discussion, in Fig. 1 we have partitioned the strategy performance space into 0 = "bad / unsatisfactory" and 1 = "good / satisfactory".

	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$	$f_7(x)$	$f_8(x)$
x_1	1	1	1	1	0	0	0	0
x_2	1	1	0	0	1	1	0	0
x_3	1	0	1	0	1	0	1	0

FIG. 1. EXAMPLE F-MATRIX.

Now let us make another reasonable assumption.

- *Uncertain World Assumption.* With complex real-world systems there are invariably certain things that we cannot measure or control, e.g., the idiosyncrasies of human behavior. These are generally captured using probability distributions, which leads to a stochastic model and stochastic optimization criteria, e.g., the average delay between submitting a request and getting a reply.

The *Uncertain World Assumption* is captured by defining a distribution over the columns of the F-matrix. That is, some problem instances (columns) are more likely than others. Conceptually then the usual goal of stochastic optimization is to find a solution $x \in X$ (row of the F-matrix) that gives only "good" performances for the most likely problem instances (columns of F). In other words, stochastic optimization can be viewed as requiring the minimization of a weighted sum of the columns of an F matrix, where the weightings are the probabilities associated with the various problem instances (columns of F).

Stochastic optimization problems of any degree of sophistication will usually require that we employ some sort of iterative search procedure to find such a solution. We can make the following reasonable assumption regarding the computational resources available to such a search procedure.

- *Intractability Assumption*. Practical constraints on computing capabilities and the time we can allocate to solving a problem limits us to procedures whose computational burden scales as a polynomial function of the size of the input, as measured by the total number of possible solutions $|X|$. If the computational effort scales faster than polynomial in $|X|$, then the time required for the procedure to complete will quickly exceed any reasonable computing and time budget.

The *Intractability Assumption* is related to the computational complexity notion that a tractable algorithm is one whose running time scales as a polynomial function of the input size (cf. Du and Ko, 2000). In computational complexity, the input size is usually measured by the number of bits needed to represent an input $x \in X$. Since $|X|$ is generally exponential in the number of bits for each input x (e.g., n-bits for each x implies $|X| = 2^n$), the Intractability Assumption is actually a relaxation of the usual computational complexity notion of intractability. In particular, under the Intractability Assumption, any deterministic problem instance $f \in F$ is assumed solvable since the effort to do so scales linearly in $|X|$. When translated to the stochastic case, which we have captured by defining a distribution over a family of deterministic problem instances, the Intractability Assumption implies that we can only optimize with respect to a number of deterministic problem instances (columns of the F-matrix) that scales polynomially in $|X|$.

Measuring Complexity. Given the *Intractability Assumption*, the total number of inputs $|X|$ is the most natural way to measure the complexity of a stochastic optimization problem. The larger $|X|$ the more complex the problem in the sense that it will generally be more difficult to obtain a satisfactory solution, i.e., one that gives good performance for all of the most likely problem instances. Moreover, $|X|$ is an optimistic lower-bound measure of complexity in that is assumes there exists polynomial time solution procedures for every deterministic problem instance (column), including the worst-case hardest instance.

The F-matrix described above gives a framework for asking general questions about optimization, complexity, and security. In particular the F-matrix can be used to explain fundamental limits to our ability to optimize and secure complex systems, and to identify guidelines for overcoming these limits. An appealing feature of the F-matrix is that the explanations it provides are simple and accessible. In fact, the essential concepts can be understood using little more than the simple example in Fig. 1 (for a rigourous development of the results see Ho, Zhao, and Pepyne, 2003; Ho and Pepyne, 2002).

Result #1: The No Free Lunch Theorem (NFLT). The NFLT [first proved for optimization in (Wolpert and Macready, 1997)] is a fundamental result in optimization, describing as it does an inverse relationship between the efficiency (speed with which it finds good solutions) of an optimization strategy and its generality (range of problems to which it can be applied). The F-matrix—even the extremely simple one in Fig. 1—makes this fundamental result clear. First, note that each row has an equal number of 0's and an equal number of 1's (this is a general property of F-matrices that holds for any $|Y|$). This implies that all row averages are equal. In other words, there is no general-purpose optimization strategy that is universally better than all others on all possible problem instances, since when averaged over all possible problem instances all strategies give equal average performance—this, in a nutshell, is the NFLT.

Result #2: Overcoming the NFLT. Not only does the F-matrix make the No Free Lunch Theorem (NFLT) clear, but it also makes it clear what we need to do to overcome it. Referring again to Fig. 1, it is clear that over *subsets* of the columns of the F-matrix, the row averages are generally *not* equal. In other words, for *specific* problems (defined by a distribution over some *subset* of the columns of F), the efficiency of different optimization techniques (each row of F is a different technique) is generally *not equal.* The goal of the decision sciences is identifying specific properties of problems and discovering optimization strategies able to exploit those specific properties, e.g., gradient methods for convex problems and the Ricatti solution to Linear Quadratic Gaussian problems.

Result #3: Simple Strategies. With outcome 0 as "unsatisfactory" efficiency and 1 as "satisfactory" efficiency, a less obvious result of the NFLT that becomes apparent when viewed through the F-matrix, is that *any* optimization strategy—no matter how simple—is good for ½ of all possible problem instances. This follows immediately from the fact that in every row ½ of the outcomes are 1's (i.e., satisfactory). Empirically, there are many simple "rules of thumb" that work well over a variety of complicated situations (e.g., tit-for-tat in the prisoner's dilemma). An interesting research direction would be to develop a means for systematically discovering simple rules of thumb and the classes of problems that they are good for (this is related to Wolfram's (2002) "new kind of science", which is based on the observation that simple programs are capable of universal computation, and that hence nothing more than simple programs are needed to understand the apparent complexity of nature).

Result #4: Robust yet Fragile. According to the *Intractability Assumption,* we can only optimize over a number of problem instances, $|P|$, that scales as a polynomial function of $|X|$. We call these $|P|$ problem instances the "planned for" problem instances. The remaining $|F| - |P|$ problem instances we call the "unplanned for" problem instances. We observe that the number of problem instances that we cannot plan for grows exponentially in $|X|$ (according to $|F| - |P| = |Y|^{|X|} - |P| \approx |Y|^{|X|}$ for large $|X|$). This observation leads to a fundamental limitation in complex systems—the unavoidable tradeoff between robustness and fragility (see also Carlson and Doyle, 2000). In particular, when we "solve" a stochastic optimization problem we do so by choosing a solution that "concentrates" good outcomes under the $|P|$ *planned for* columns. Then, simply because there is an equal number of good and bad outcomes in each row, the *unplanned for* columns will necessarily have a higher fraction of bad outcomes. Hence a solution that is "robust" in that it gives good outcome for all of the $|P|$ planned for columns, is "fragile" in that it is more likely than not to give bad outcome when an unplanned for situation is encountered. Moreover, this fragility becomes more apparent as the complexity of the problem $|X|$ increases, since as $|X|$ grows, the fraction of problem instances that we can plan for decreases at an exponential rate, i.e., $|P| / (|F| - |P|) \approx 1/|Y|^{|X|}$. In social, legal, and political systems, which are among some of the most complex human "engineered" systems, this observation is the oft-heard "law of unintended consequences" (Merton, 1936).

Result #5: Challenge of Security. One common way to view security is as a matrix game in which a defender (security administrator) chooses a defense strategy (a row from the F-matrix), an adversary chooses an attack scenario (column from the F-matrix), and the payoff is the corresponding F-matrix entry (0 = bad outcome for the defender, 1 = good outcome for the defender). However, because every row contains an equal number of 0's and 1's, it is clear that *no* security strategy can protect against every *possible* attack. And similar to the robust yet fragile tradeoff, the likelihood that our system will have a security vulnerability increases exponentially with $|X|$, the system complexity.

The moral lessons that we can take away from the above is that we face certain unavoidable limitations in our problem solving capabilities—there is no Holy Grail solution procedure and sometimes obtaining the optimal solution that gives good outcome for every likely problem instance is simply beyond our capabilities. This does not mean, however, that we should give up. There are some design guidelines that we can follow. The first and most obvious solution is to try to reduce problem complexity $|X|$. The most obvious way to do this is by removing certain features and capabilities, e.g., prevent remote login or wireless access to a sensitive information network. Another way to reduce complexity is by imposing more structure on a problem. In a network this might be done, for example, by replacing a "flat" interconnection topology, where there are many paths between the network nodes, with a hierarchical one where paths are clearly defined and more easily monitored and controlled; or we might host our Web pages on a special server that is isolated from the rest of our internal organizational network. Reducing features and building in structure effectively reduces the number of columns in the F-matrix, i.e., starting with the original problem we end up making certain problem instances (columns) infeasible. Another way to eliminate columns of the F-matrix is to make them computationally intractable. For example we might require that all communications be encrypted to complicate passive eavesdropping. Imposing structure and reducing features may reduce system performance, but this might be more than compensated by a reduction in security vulnerabilities and other unintended consequences. Finally, we can deal with complexity by relaxing our optimization goal, i.e., rather than asking for the true (generally unattainable) optimal, we might instead ask for solutions that are "good enough with high probability" or simply "improving". Good enough with high probability greatly expands the number of solutions that will satisfy our objectives [goal relaxation is the essence of the ordinal optimization approach (Ho, 1999)]. The idea of seeking improving solutions is that while the optimal may be computationally out of reach, comparing solutions and picking the better one is computationally easier [order is easier to determine than value, (Ho, 1999)], and when pursued over a period of time can give steady gains, even as objectives and the environment change over time. The Internet for example continues to grow in size and importance at the same time that its use continues to evolve with each new technological innovation.

Possible vs. Probable. As suggested above, a major question in dealing with complexity theory is not so much what is *possible* but rather what is *probable.* As the complexity of a system increases, what is possible quickly exceeds our abilities to contemplate (what is possible grows exponentially in $|X|$). However, there is currently no theory—not F-matrix theory or any other theory—that can reliably tell us what is *probable* (i.e., what columns are most likely in specific settings). Empirical studies of complex networked systems show us that certain features such as small world and scale free interaction topologies appear again and again (Watts, 2003; Barabasi, 2002). Thus, it seems that there may be some underlying rules at work that "prune" the space of the possible to a much smaller space of the probable (see also Morowitz, 2002). The probable vs. the possible remains an open research question.

2.2 Cascading Failure

In (Pepyne, Gong, and Ho, 2001) we developed a modeling framework for studying how malicious software such as Internet worms and email viruses spread. The framework is derived from the mathematical machinery of stochastic branching models and percolation models from statistical physics. The intent of the modeling framework is vulnerability analysis—to understand the factors that determine how Internet worms and email viruses spread so that methods for early detection and mitigation might be developed.

In our model we represent the physical Internet as a graph with "edge" nodes for the servers, workstations, and local area networks, and "internal" nodes for the routers, and gateways. The links connecting the nodes represent the physical communication links, e.g., wires, optical fibers, and wireless channels.

On top of the physical Internet infrastructure are a number of "logical overlay" networks. One such logical network is defined by the TCP/IP protocol suite. In this logical network each node is an IP address. The way TCP/IP works, it makes it appear as if every IP address has a direct connection to every other IP address. That is, the logical network defined by the TCP/IP protocols is a complete graph like the one shown in Fig. 2; the actual convoluted path that the messages follow through the physical network is hidden by the protocols.

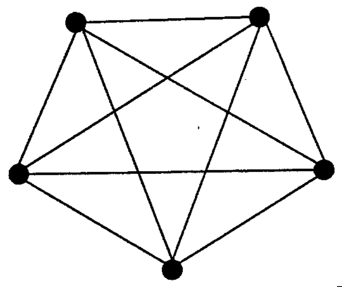
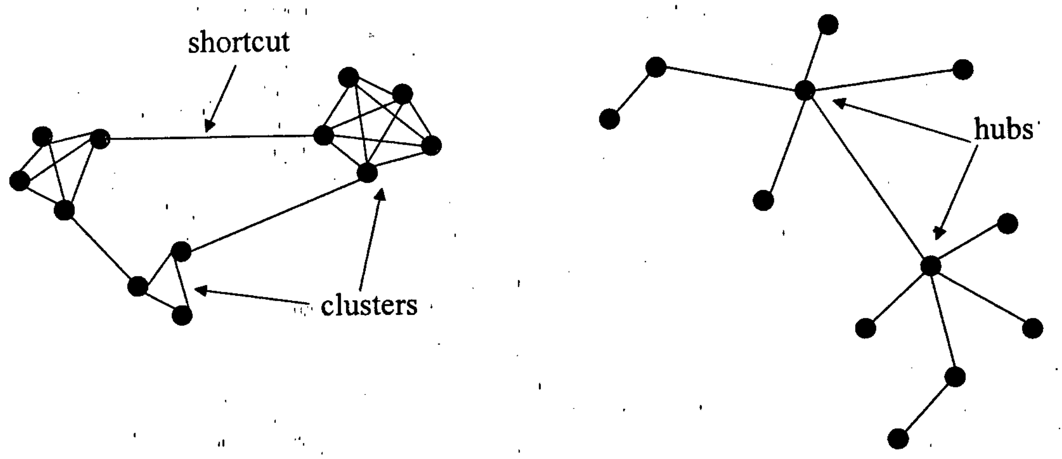


FIG. 2. LOGICAL INTERCONNECTION TOPOLOGY BETWEEN IP ADDRESSES.

Services such as email, chat, instant messaging, and the way Web pages are linked together define other logical networks overlaid on top of the physical Internet infrastructure. The interconnection topologies of these logical networks is defined by email address books, chat and instant messaging buddy lists, and Web page hyperlinks. These networks are "social" networks formed between friends, work mates, business associates, and related Web pages. Recent research into such networks shows that they have special topological characteristics termed "small world" and "scale free"[1] (Watts, 2003; Barabasi, 2002). What is special about these

[1] The graph is called scale free because the distribution of the number of edges connected to each node follows a power law distribution. Power law distributions have no natural scale; their distributions give a straight line on a log-log plot. The "tail" of a distribution with natural scale, in contrast, will sharply drop off on a log-log plot.

topologies is that even though they are sparse[2] the average number of hops between any pair of arbitrarily selected nodes is surprisingly low.[3] In particular, of the billions of Web pages it is estimated that there are an average of only 19 clicks between any two arbitrarily selected Web pages, and of the 6 billion humans on the planet there is said to be only "six degrees of separation" between any arbitrarily selected pair (Watts, 2003; Barabasi, 2002). Both of these network topologies can be viewed as hierarchical—small world networks as a hierarchy of "clusters" interconnected by a few "shortcut" links, and scale free networks as a hierarchy of highly connected "hubs" each surrounded by a large constellation of lightly connected nodes. Fig. 3 schematically illustrates these two topologies.



(a) Small World (b) Scale Free

FIG. 3. SMALL WORLD AND SCALE FREE NETWORKS.

Our spreading model begins with a graph of the appropriate topology—a complete graph for Internet worms which spread from IP address to IP address, and a small world or scale free graph for email, chat, instant messaging, or malicious Web page scripts. At each node of the graph is a finite state machine (FSM). As the malicious program spreads along the network's edges it changes the states of the FSMs from a not infected (normal) state through to an infected (compromised) state. FSM state transitions depend both on internal parameters (email viruses for example generally need a human user to activate the virus) and the states of its neighbors (malicious programs spreads by hopping from neighbor to neighbor).

Internet worms

An Internet worm, such as the CodeRed and NIMDA worms of 2001 and the Slammer worm of 2003, connects to a web server and automatically replicates itself. Worms are fully automated mobile programs that require no human intervention to spread. The basic spreading mechanism for a worm consists of the following three steps: (1) the worm randomly scans IP addresses looking for servers to infect; (2) when the worm finds another server it "probes" it to see if it is vulnerable; and (3) if the probed server is vulnerable, the worm installs and runs a copy of itself

[2] With N the number of nodes in the graph, *sparse* means that the number of links in the graph is only a small fraction of the N(N−1)/2 total number of possible links that the graph would have if it were completely connected.
[3] The number of hops averaged over all pairs of nodes is called the graph's *diameter*.

there. In this way, the number of worms scanning and trying to infect vulnerable servers multiples. The basic spreading model for a worm is shown in Fig. 4. As illustrated in the figure, to a worm the Internet looks like a complete graph, where each node of the graph is an IP address. In its simplest form, each IP address has two states: *uninfected* (state 0) and *infected* (state 1). State transition between the uninfected and infected states is given by $q_i(k)$, which is the probability that node i will become infected at step k; once a node becomes infected it stays infected in this simple model (for more detailed models, see Zou, Gong, and Towsley, 2002). The probability $q_i(k)$ depends on whether or not node i is scanned at time k and whether or not node i is vulnerable. Assuming random scanning, the usual method, the probability node i is scanned is independent of whether or not it is vulnerable, and depends only on the total population of infected nodes at time k. With random scanning, the probability node i is vulnerable is equivalent to the fraction of IP addresses (there are 2^{32} in IPv4) that are vulnerable. Once infected the worm continuously scans for new victims.

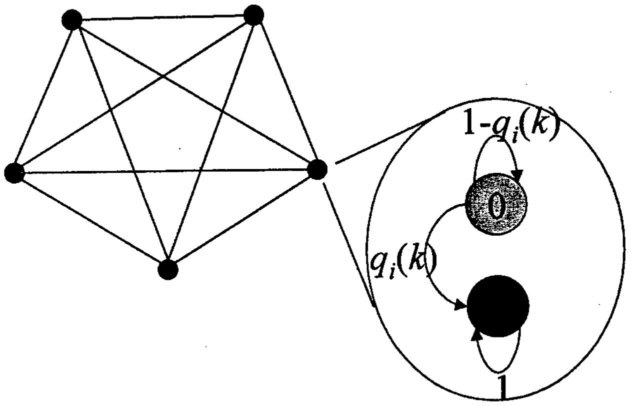


FIG. 4. BASIC SPREADING MODEL FOR INTERNET WORMS.

Email viruses

The most famous email virus is probably the Love Bug virus, which made its way around the world in 1999. The typical email virus is an email message containing a virus program as an attachment. Unlike an Internet worm, which is fully automatic, email viruses require human assistance; a human must often "open" the attachment in order to activate the virus. The basic spreading mechanism for an email virus consists of the following steps: (1) an email message with a virus attachment is sent; (2) the recipient reads the email message and opens the attachment; (3) opening the attachment runs the virus; (4) if the recipients email client program (e.g., Eudora, Outlook) is vulnerable to the virus, the virus infects it; (5) the first thing the virus does is attempt to spread by emailing a copy of itself to every name found in the email program's address book and / or inbox; (6) the virus then goes to "sleep" (becomes dormant) or its malicious part is activated to do its damage.

The basic spreading model for an email virus is shown in Fig. 5. To an email virus, the Internet looks like a small world or scale free topology, where each node is an email address that it knows (e.g., one in its inbox or in its address book). The simplest form for an email virus has three states, an uninfected state (state 0) an infected and broadcasting state (state 1) and a dormant or malicious state (state 2). A state transition between the 0 state and the 1 state occurs with probability $q_i(k)$, which is the probability that the email client is vulnerable, has received an email virus, and the virus has been "opened" by the human user. While in state 1 the virus sends a copy of itself to all of its neighbors. The virus then immediately transitions to state 2 where it either becomes dormant or runs its malicious payload (e.g., crashes the computer, steals files, installs a Trojan, etc.). A more detailed study of email viruses can be found in (Zou, Towsley, and Gong, 2002).

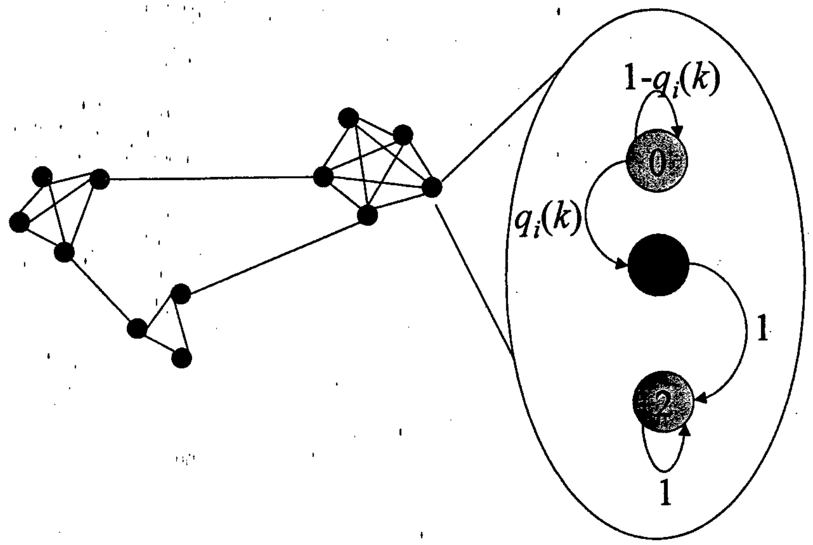


FIG. 5. BASIC SPREADING MODEL FOR EMAIL VIRUSES.

As described above, there are three basic differences between worms and email viruses. First, worms are fully automatic, while email viruses typically require human assistance. Second, worms continue to seek out victims one at a time until the infected server is shut down, while email viruses do it with a single mass mailing. Third, worms spread over a complete graph, while email viruses spread over a small world or scale free graph topology.

Spreading dynamics

Despite the differences in their spreading mechanisms, the general dynamics for worms and email viruses is the same—the *S*-shaped logistic growth curve shown in Fig. 6. Starting from a small pool of infected nodes, the number of infected nodes suddenly takes off at an exponential rate, tailing off after some fraction of the nodes have been infected. In the case of worms, 100% of the vulnerable nodes will become infected, and this is true for any infection probability q_i (smaller q_i, of course, result in a slower spreading rate). For email viruses, the fraction of nodes infected by the time the virus stops spreading depends both on q_i and on where the infection begins. If the virus can reach a few of the hubs (people with a huge number of email contacts), then the chance of the infection reaching "epidemic" stage or crossing "percolation" threshold is increased. These hubs in the scale free graph are its "Achilles heal" when it comes to the graphs' fault tolerance and security (see also Albert, Jeong, and Barabasi, 2000; Barabasi, 2002).

Empirically the Code Red worm in 2001 saturated the population of vulnerable servers in less than 24 hours, the 2003 Slammer worm in 10s of minutes. It is said that the Love Bug email virus traveled around the world in about 6 hours. Clearly, worms and email viruses spread much to fast for manual intervention to be of much help. Moreover, there is often no way to tell if a connection request at a server is a worm or an attachment is an email virus until the request is allowed or the attachment opened. And even then the server may not be able to detect that it is infected; similarly a human opening an attachment may not notice anything unusual either.

Given that worms and viruses can spread much faster than can be controlled by manual intervention, the idea then is to try to develop an early warning system that can detect worm or email virus activity before the spread reaches the "knee" of the exponential growth phase. At this point servers and email clients generating suspicious traffic could be "quarantined" and traffic through certain critical points in the network could be "filtered" out and held as "guilty until proven innocent." For some work in this direction for worm mitigation see (Zou, et. al, 2003). In that work, exponential growth is used as the "signature" of widespread worm attack.

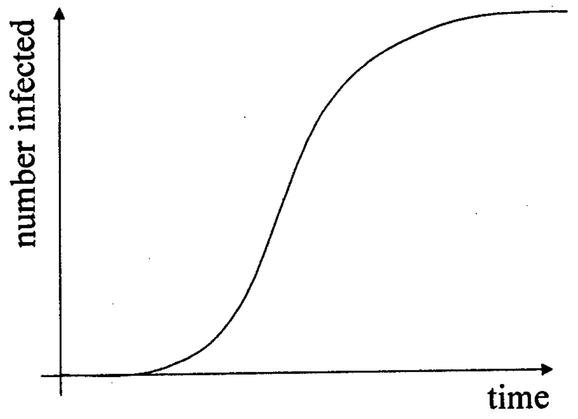


FIG. 6. BASIC SPREADING DYNAMICS FOR WORMS AND VIRUSES.

2.3 Information Classification

Organizations today make virtually all of their information assets available via their organizational networks. If these networks are accessible from the Internet, then these information assets are exposed to being attacked by *outsiders*. Outsiders typically break into a network by masquerading as an authorized user whose credentials (e.g., username and password) they have either guessed or stolen. This is not hard to do as usernames are not considered secret and passwords are notoriously weak; they can be relatively easily obtained using password cracking software, sniffed off insecure wireless links, stolen with Trojan horses that monitor keystrokes, or sometimes obtained with a phone call and a little social engineering. But quite possibly worse than outsiders gaining access, however, is the damage that malicious *insiders* can do. Unlike the typical outsider, insiders generally know a great deal about organization and the organization's network, including what assets there are and how to find or disrupt them.

Detecting break-ins by outsiders is the goal of an *intrusion detection system* (IDS). Detecting malicious insiders is the goal of a *misuse detection system* (MDS). The "sensors" used by these systems are classified as *signature detectors* that look for known patterns of intrusion or misuse, and *anomaly detectors* that look for deviations from historical patterns of behavior. Any comprehensive detection system needs both types of detectors. Signature detectors are typically fast but not good at detecting new kinds of attacks, while anomaly detectors are generally slower but can detect new kinds attacks.

The basic premise of intrusion and misuse detection is that the "behavior" associated with normal, benign activities can be distinguished from that of masquerading outsiders and malicious insiders. On computers, behavior might be how a person interacts with a keyboard or mouse, which computer a person uses, what application programs they are using, etc. But regardless of what behavior is monitored, intrusion and misuse detection is at it heart a problem in feature selection—what features will best allow us to detect intrusions and computer misuse? Since ultimately all behaviors on computers generate operating system calls, a majority of intrusion and misuse detection systems focus on *operating system calls* and *system call sequences*. Taking a different approach, in (Hu, Pepyne, and Gong, 2002) we examined the potential detection capabilities of simple temporal features, i.e., instead of attempting to classify users by the content of their "conversations" with a computer (e.g., by system call analysis), our objective was to attempt to classify them by the *timing* and *rhythms* of the conversations (i.e., by temporal analysis).

People exhibit temporal regularities at all levels, from the sleep / wake cycle, brain waves, heart beat, muscle tone, to how we walk, write, and type on a keyboard. Law enforcement is already well aware of these temporal regularities and routinely use them to detect deception when conducting interrogation, polygraph tests, and for profiling (certain activities OK during the day, are quite suspicious at night). Our purpose was to see if similar ideas could profile users to (1) distinguish between different people, and (2) detect changes in the peoples' behaviors.

We used 6 features: (1) interval – time elapsed since last log on, (2) length – duration of session, (3) output – total number of commands issued during the session, (4) density – mean command rate, (5) timing.day – the day of the week the session started, and (6) timing.hour – the hour of the day the session started. Each time a user logged on and off the computer, one such 6-dimensional data point was generated. Results from a population of graduate engineering students (at the University of Massachusetts, Amherst) showed that even with only these 6 simple features clear differences could be detected between different users; differences clear even to the naked eye.

Intrusion and misuse detection is a classification problem—either the observed behavior is not anomalous (null hypothesis) or it is (alternate hypothesis). Using the 6 temporal features described above, we used the method of logistic regression to build a classifier system. Logistic regression, which can be viewed as a statistically rigorous way to train a neural network classifier, is a technique that has not been widely used for intrusion and misuse detection. While we admit that our results were not outstanding, they did provide us with a clear proof of concept that simple temporal features such as the most basic ones that we selected can be effectively used to distinguish between different users and to detect changes in a user's behavior over time.

In a society concerned about privacy, a major advantage of our approach over others is that it is minimally invasive, and privacy preserving, requiring only simple time stamping and counting. Our approach does not collect detailed command histories during its analysis or build an audit file of commands issued (itself subject to attack). Its limitation, like all methods for anomaly detection is that it is not generally capable of real-time detection. However, our results did clearly establish a clear role for temporal features as a component part of any comprehensive defense in depth computer network security solution.

2.4 Technology Vulnerabilities

Increasingly, wired networks are being expanded and enhanced by adding wireless access for the cost savings and for the flexibility and convenience that mobility offers. Whereas wired data links physically provide a well-defined and protected data link between devices, the problem with wireless links is that wireless signals are broadcast over a wide area, which makes wireless data links inherently insecure. To attack a wired data link generally requires physical access to the wire. Attacking a wireless data link, in contrast, can be done from an organization's parking lot by anyone with a laptop and a wireless networking card.

Making wireless data links as secure as wired ones means first that all messages traveling over wireless data links must be authenticated, and second it means that all messages must be encrypted. Technically speaking this is not a particularly challenging problem, and there are well-developed cryptographic primitives capable of providing a solution. However, any cryptographic solution is only as secure as its implementation, and when improperly implemented even the most theoretically secure scheme can be easily compromised. The original *Wired Equivalent Privacy* (WEP) security protocol used in the IEEE 802.11b wireless communication standard is a perfect case in point (Karygiannis and Owens, 2002; Petroni and Arbaugh, 2003). Wireless data links "secured" with WEP can be compromised in under 30 minutes using virtually any standard laptop computer and "cracking" tools such as "AirSnort", which is easily found on-line. Since 802.11b is still the dominant standard (in the U.S. at least) and is very widely deployed, its insecurity leaves a huge number of personal, academic, government, and businesses computers and computer networks virtually wide open to attack.

The main problem with WEP is that the secret keys used for message encryption are rarely (often never) changed. This combined with the use of a stream cipher makes it almost trivial to build a dictionary of key sequences. For all practical purposes, this is equivalent to determining the secret key, since given such a dictionary all communications are compromised. In (Pepyne, Ho, and Zheng, 2003) we developed a simple replacement for WEP that we call SPRiNG, short for *Synchronized Pseudo-Random Number Generation* security protocol. SPRiNG corrects WEP's problem by using a different encryption key for each and every message sent. Specifically, both parties in a unicast session, e.g., a laptop and a network access point (a.k.a. a wireless base station), have identical pseudo-random number generators (PRNG). These PRNGs are synchronized at the start of the communication session by giving each party a common random seed. With this seed, both parties can individually generate the exact same sequence of pseudo random numbers (PRNs), $\{R_1, R_2, \dots\}$. Without knowledge of the seed, which is a secret shared only by the two communicating parties, no one else can correctly generate this sequence. Then as the two parties communicate, they encrypt the kth message exchanged between them with the kth PRN, R_k, in the PRNG sequence. Since both parties must keep count of the number of messages sent, k, and since frame loss in wireless is not infrequent, we employed a simple K-look ahead window to keep them synchronized (with K determined by the expected frame loss probability). This simple scheme just described provides *message authentication* (because the PRN sequence is almost random and hard to guess), *replay protection* (because the PRN sequence almost never repeats), and *confidentiality* (via encryption with a different encryption key applied to each message).

SPRiNG's security largely depends on keeping the PRNG seed secret from attackers. An attacker who can determine the seed can reproduce the PRN sequence, and would consequently be able to decrypt confidential messages and insert properly encrypted forged messages at will. In general, "inverting" a PRNG to determine its seed is not entirely trivial even when the raw PRNG sequence, $\{R_1, R_2,\ldots\}$, can be directly observed. But to make is almost impossible, the SPRiNG protocol never reveals the raw PRN sequence to would be attackers. A SPRiNG message contains only the sender's value for the message count, k, and the message itself encrypted with R_k. To get the PRNG seed, therefore, an attacker would first have to "invert" the stream cipher to get the R_k's. Then the attacker would have to invert the PRNG to get the seed. We moreover propose that the secret seed should be a different random number separately renegotiated at the beginning of each communication session and whenever a mobile wireless device roams from the coverage area of one access point to the coverage area of another (i.e., during access point handoff). Session seed negotiation could be effected, for example, using a "master" secret key for each registered wireless device along with one of any number of existing key negotiation protocols. (We are also exploring a version of SPRiNG that would perform key updating using a *covert channel* effected by "skipping" and "permuting" elements of the PRN sequence in a way that the intended recipient can detect but an attacker cannot. We omit that discussion here as its security analysis was not complete at the time this report was written.)

Since our intention was to use SPRiNG as a replacement for WEP, we designed an implementation that is the same as WEP in every way (so we could use WEP's encryption and integrity checking mechanisms, which are done in hardware) except for the following differences. The addition of a PRNG generator (a computationally simple linear congruential generator (LCG) will suffice), a simple look ahead window mechanism to keep the sender and receiver synchronized, and a table in the wireless access points for storing the count k and session seed for each active wireless session. While others are calling for extensive and complicated changes, we argue that with only these few simple changes wireless security could be dramatically improved.

In closing, we remark that although we developed SPRiNG to deal with the current and pressing need for improved wireless security, the SPRiNG protocol is really a generic protocol that can be used anywhere that secure point to point communication is required. Potential applications include, for example, secure communication between a computer and peripherals such as keyboards, printers, external storage devices; secure communication between routers as part of a "hardened" routing infrastructure; and even to secure communications between applications and processes.

3. LIST OF PUBLICATIONS

The following is a list of the publications completed during the period of this project. These publications can be made available upon request.

1. **D.L. Pepyne**, W.-B. Gong, and Y.-C. Ho (2001), "Modeling and Simulation for Network Vulnerability Assessment," presented at the 40th *U.S. Army Operations Research Symposium* (AORS), Fort Lee, VA.

2. Y.-C. Ho and **D.L. Pepyne** (2002), "Simple Explanation of the No-Free-Lunch Theorem and Its Implications," *Journal of Optimization Theory and Applications*, Vol. 115, No. 3, pp. 549-570.

3. Y.-C. Ho, Q.-C. Zhao, and **D.L. Pepyne** (2003), "The No Free Lunch Theorems, Complexity, and Security," *IEEE Transactions on Automatic Control*, Vol. 48, No. 5, pp. 783-793.

4. J.-H. Hu, **D.L. Pepyne**, and W.-B. Gong (2002), "User Classification by Logistic Regression on Temporal Behaviors," presented at the 41st *U.S. Army Operations Research Symposium* (AORS), Fort Lee, VA.

5. **D.L. Pepyne**, Y.-C. Ho, and Q.-H. Zheng (2003), "SPRiNG: Synchronized Random Numbers for Wireless Security," *Proceedings of the IEEE Wireless Communications and Networking Conference* (WCNC'03), New Orleans, LA.

4. FELLOWS' BIOGRAPHICAL SKETCH

The following is a brief biographical sketch of the fellow funded by this fellowship program.

Name and Contact Information (as of July 2003)

Dr. David L. Pepyne

Research Associate

Division of Engineering and Applied Sciences
Harvard University

Maxwell Dworkin, Room 149
33 Oxford Street
Cambridge, MA 02138
PHONE: 617-495-8911; FAX: 617-495-9837
EMAIL: *pepyne@hrl.harvard.edu*; WWW: *hrl.harvard.edu/~pepyne*

Background

Education

1999	Ph.D., *Electrical and Computer Engineering*, University of Massachusetts, Amherst, MA.
1995	Masters, *Electrical and Computer Engineering*, University of Massachusetts, Amherst, MA.
1986	Bachelors, *Electrical Engineering*, University of Hartford, CT.

Experience

1999-	*Research Fellow*, Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA.
1995-1997	*Project Engineer*, Alphatech, Inc., Burlington, MA.
1991-1999	*Research Assistant*, various departments, University of Massachusetts, Amherst, MA.
1986-1990	*Military Officer*, United States Air Force, Flight Test Engineer, Edwards AFB, Edwards, CA.
1983-1986	*Electronic Technician*, Buckland G-Line Cable Television, Buckland, MA.
1983-1986	*Police Officer*, Town of Buckland, Buckland, MA.

Current Position and Future Plans

In the near term, Dr. Pepyne will continue as a Research Associate at Harvard doing research focusing on theoretical and practical issues related to optimization, complexity, robustness, and security. Longer-term plans include a research position within academia and / or the formation of a consulting and research company.

5. REFERENCES

The following lists the supplementary references used in this report.

R. Albert, H. Jeong, and A. Barabasi (2000), "Error and Attack Tolerance of Complex Networks," *Nature*, Vol. 406.

A.-L. Barabasi (2002), *Linked: The New Science of Networks*, Perseus Publishing.

J. M. Carlson and J. Doyle (2000), "Highly Optimized Tolerance: Robustness and Design in Complex Systems," *Physical Review Letters*, Vol. 84, No. 11, pp. 2529-2532.

D.-Z. Du and K.-I. Ko (2000), *Theory of Computational Complexity*, Wiley-Interscience.

Y.-C. Ho (1999), "An Explanation of Ordinal Optimization: Soft Computing for Hard Problems," Information Sciences, Vol. 113, pp. 169-192.

T. Karygiannis and L. Owens (2002), "Wireless Network Security: 802.11, Bluetooth, and Handheld Devices," NIST Special Publication 800-48, DRAFT, 24 July 2002, csrc.nist.gov/publications/drafts.html.

S. Lloyd (2001), "Measures of Complexity: A Nonexhaustive List," *IEEE Control Systems Magazine*, Vol. 21, No. 4, pp. 7-8.

R. K. Merton (1936), "The Unanticipated Consequences of Purposive Social Action," *American Sociological Review*, Vol. 1, No. 6.

H. J. Morowitz (2002), *The Emergence of Everything: How the World Became Complex*, Oxford University Press.

N.L. Petroni, Jr. and W.A. Arbaugh (2003), "The Dangers of Mitigating Security Design Flaws: A Wireless Case Study," *IEEE Security and Privacy*, Vol. 1, No. 1, pp. 28-36.

D. J. Watts (2003), *Six Degrees: The Science of the Connected Age*, W.W. Norton & Company.

D. H. Wolpert and W. G. Macready (1997), "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 67-82.

C-.C. Zou, L. Gao, W.-B. Gong, and D. Towsley (2003), "Monitoring and Early Warning for Internet Worms," *University of Massachusetts, Electrical and Computer Engineering Technical Report* TR-CSE-03-01.

C. Zou, W. Gong, and D. Towsley (2002), "Code Red Worm Propagation Modeling and Analysis," *Proceedings of CCS'02*, Washington D.C., November.

C. Zou, W. Gong, and D. Towsley (2002), "Email Virus Propagation Modeling and Analysis," tennis.ecs.umass.edu/~czou/research/emailvirus.pdf.